



Міністерство освіти і науки України
Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Лабораторна робота №8

Технології розроблення програмного забезпечення
Тема: Патерни проектування

Виконав

студент групи IA-34:

Жительний В.В.

Перевірив:

Мягкий М.Ю.

Мета: Вивчити структуру шаблонів «Composite», «Flyweight» (Пристосуванець), «Interpreter», «Visitor» та навчитися застосовувати їх в реалізації програмної системи.

Тема:

21. **Online radio station** (iterator, adapter, factory method, facade, visitor, client-server)

Додаток повинен служити сервером для радіостанції з можливістю мовлення на радіостанцію (64, 92, 128, 196, 224 kb/s) в потоковому режимі; вести облік підключених користувачів і статистику відвідувань і прослуховувань; налаштовувати папки з піснями і можливість вести списки програвання або playlists (не відтворювати всі пісні).

Хід роботи

Необхідно було реалізувати функціонал генерації звіту, який потребує обходу ієрархічної структури об'єктів (Station -> Playlist -> Track). Замість того, щоб додавати методи логіки звіту в самі моделі, що порушило б принцип SRP, було використано Visitor. Це дозволило винести алгоритм аналізу структури в окремий клас.

Графічне представлення структури реалізації даного шаблону наведено на діаграмі класів

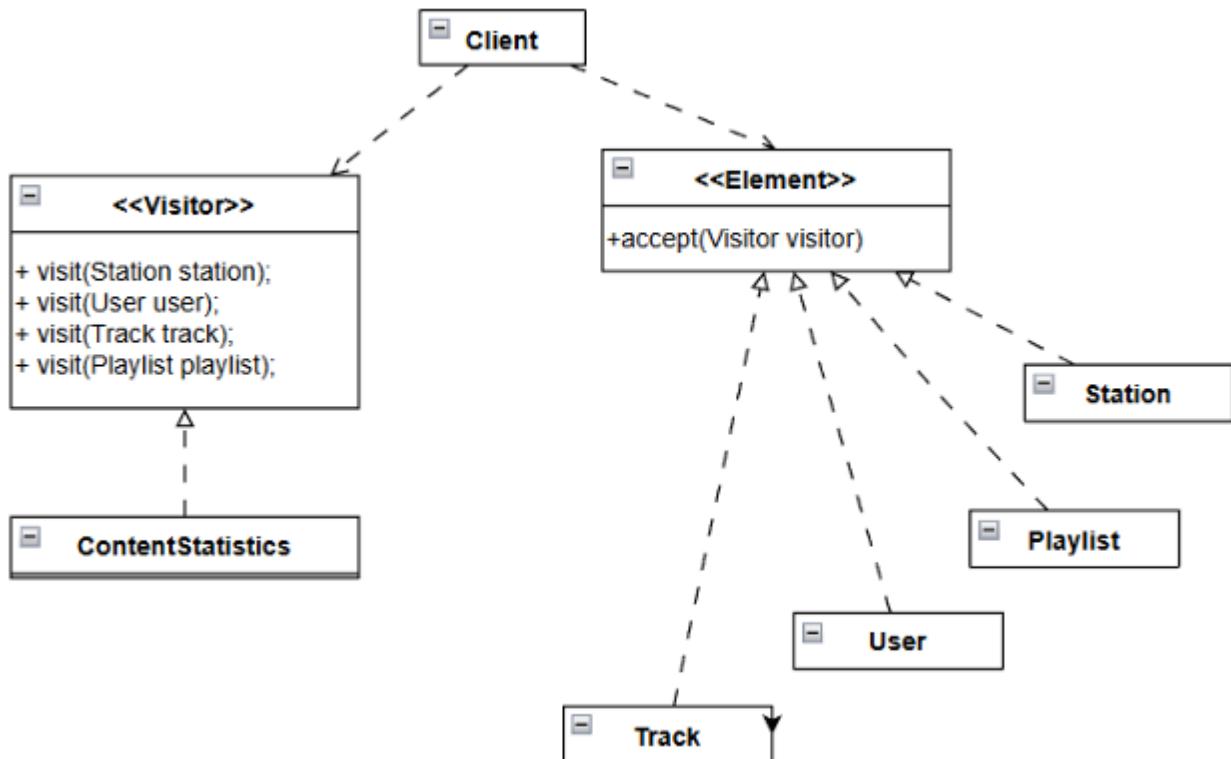


Рис. 1 Діаграма класів для шаблону Відвідувач

Діаграма ілюструє структуру патерну Відвідувач. Клієнт (Client) ініціює виконання операцій над об'єктами, взаємодіючи через абстракції. Інтерфейс Visitor оголошує методи відвідування для кожного типу сущностей (Station, User, Track, Playlist), а клас ContentStatistics реалізує цей інтерфейс, інкапсулюючи конкретну бізнес-логіку збору статистики. Сущності системи (Station, Playlist тощо) реалізують інтерфейс Element та метод accept(), який забезпечує механізм подвійної диспетчеризації, дозволяючи візитеру обробляти ці об'єкти без зміни їхнього коду.

Програмна реалізація

Інтерфейси:

```
public interface Element {  
    void accept(Visitor visitor);  
}  
  
public interface Visitor {  
    void visit(Station station);  
    void visit(User user);  
    void visit(Track track);  
    void visit(Playlist playlist);  
}
```

Реалізація інтерфейсу Visitor:

```
public class ContentStatistics implements Visitor{  
  
    private int totalTracks = 0;  
    private long totalDurationSeconds = 0;  
    private int playlistCount = 0;  
    private String stationName = "";  
  
    @Override  
    public void visit(Station station) {  
        this.stationName = station.getName();  
    }  
  
    @Override  
    public void visit(User user) {  
    }  
  
    @Override  
    public void visit(Track track) {  
        totalTracks++;  
        totalDurationSeconds += track.getDurationInSeconds();  
    }  
  
    @Override  
    public void visit(Playlist playlist) {  
        playlistCount++;  
    }  
}
```

```

    }

    public int getTotalTracks() {
        return totalTracks;
    }

    public long getTotalDurationSeconds() {
        return totalDurationSeconds;
    }

    public int getPlaylistCount() {
        return playlistCount;
    }

    public String getStationName() {
        return stationName;
    }
}

```

Деякі з реалізацій інтерфейсу Element:

```

public class Track implements Element {

    . . .

    @Override
    public void accept(Visitor visitor) {
        visitor.visit(this);
    }
}

public class Station implements Element {

    . . .

    @Override
    public void accept(Visitor visitor) {
        visitor.visit(this);

        if (playlists != null) {
            for (Playlist playlist : playlists) {
                playlist.accept(visitor);
            }
        }
    }
}

public class Playlist implements TrackCollection, Element {

    . . .

    @Override
    public void accept(Visitor visitor) {
        visitor.visit(this);

        if (tracks != null) {
            for (Track track : tracks) {
                track.accept(visitor);
            }
        }
    }
}

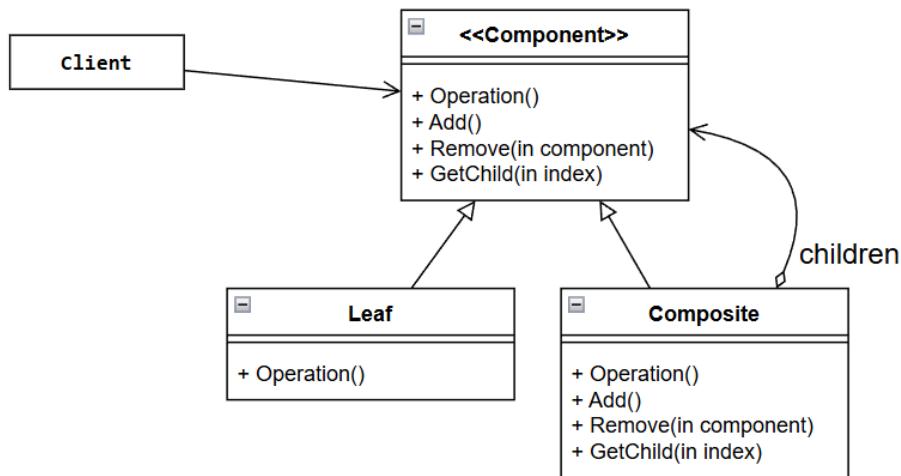
```

Контрольні питання:

1. Яке призначення шаблону «Композит»?

Використовується для складання об'єктів у деревоподібну структуру для подання ієархій типу «частина-ціле». Дозволяє обробляти поодинокі об'єкти та групи об'єктів однаково

2. Нарисуйте структуру шаблону «Композит».



3. Які класи входять в шаблон «Композит», та яка між ними взаємодія?

Component: Абстракція з операціями (Operation, Add, Remove).

Leaf: Кінцевий елемент, що виконує роботу.

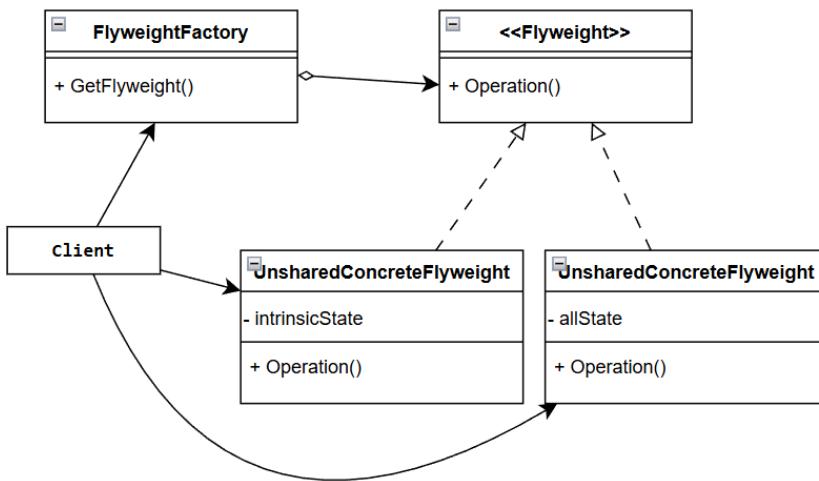
Composite: Зберігає колекцію дочірніх компонентів (`children`) і реалізує методи управління ними.

Клієнт викликає операцію у **Component**. Якщо це **Composite**, він проходить по списку своїх дітей і викликає цю операцію у них (рекурсивно).

4. Яке призначення шаблону «Легковаговик»?

Використовується для ефективної підтримки великої кількості дрібних об'єктів шляхом поділу спільних даних між ними (зменшення споживання пам'яті).

5. Нарисуйте структуру шаблону «Легковаговик».



6. Які класи входять в шаблон «Легковаговик», та яка між ними взаємодія?

FlyweightFactory: Створює та управляє пулом легковаговиків.

Flyweight: Інтерфейс, через який легковаговики можуть отримувати зовнішній стан.

ConcreteFlyweight: Реалізує інтерфейс і зберігає внутрішній стан (спільний).

Стан ділиться на внутрішній (спільний, зберігається у Flyweight) та зовнішній (контекст, передається клієнтом у метод Operation). Фабрика перевіряє, чи існує вже такий об'єкт: якщо так, то повертає його, якщо ні – створює новий .

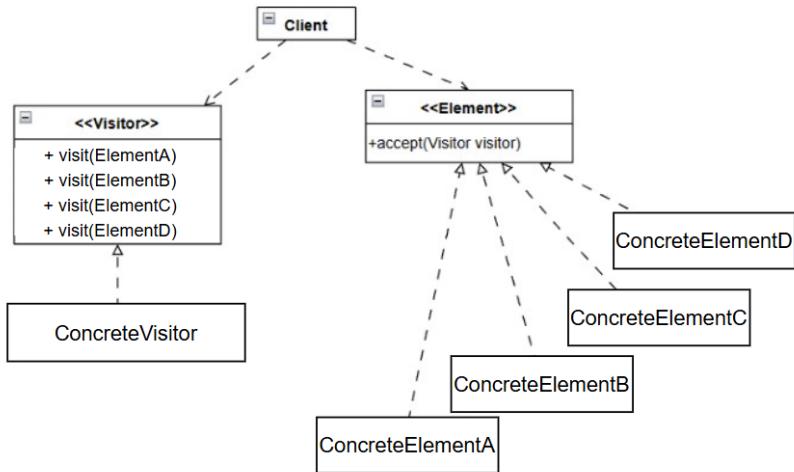
7. Яке призначення шаблону «Інтерпретатор»?

Використовується для визначення граматики простої мови та створення механізму, який інтерпретує речення цієї мови. Він найчастіше використовується для створення предметно-орієнтованих мов, парсингу математичних виразів або обробки текстових команд.

8. Яке призначення шаблону «Відвідувач»?

Дозволяє додавати нові операції над об'єктами, не змінюючи класи цих об'єктів. Відокремлює алгоритм від структури об'єкта

9. Нарисуйте структуру шаблону «Відвідувач».



10. Які класи входять в шаблон «Відвідувач», та яка між ними взаємодія?

Visitor: Інтерфейс з методами visit().

ConcreteVisitor: Реалізує операцію для кожного типу елемента.

Element: Інтерфейс з методом Accept(Visitor).

ConcreteElement: Реалізує Accept, викликаючи відповідний метод відвідувача.

Клієнт проходить по структурі об'єктів і для кожного викликає Accept(visitor). Елемент "впускає" відвідувача, викликаючи його метод, що відповідає цьому типу елемента.

Висновок: Під час виконання даної лабораторної роботи було досягнуто поставлену мету: вивчено та проаналізовано групу шаблонів проєктування, що використовуються для вирішення типових задач в об'єктно-орієнтованому програмуванні. Було детально розглянуто призначення, структуру та принципи взаємодії класів для наступних патернів: «Composite», «Flyweight» (Пристосуванець), «Interpreter», «Visitor». А також на практиці в програмному коді було реалізовано шаблон «Відвідувач».